

AMENDMENT

Please amend the above-identified application as follows:

In the Specification

Please replace the paragraph beginning at page 6, line 19 of Applicants' original specification, with the following amended paragraph:

The example of FIG. 1 includes an installation package (312) for installation on a computer system by installation application (102). An installation package is a collection of files, data files and executables, organized specially for ease of installation by use of an installation application. Installation application both package files for installation and install the packages. Examples of installation applications include the 'Red Hat Package Manager_{TM}' ("RPM") and IBM's InstallShield MutliPlatform_{TM}. ("ISMP"). Installation applications are typically capable of installation, uninstallation, verification, query support, and updates through a command line interface or an application programming interface ("API"). An installation package may organize files for installation according to products, bundles, filesets, and so on, and, installation package (312) also includes workload management properties (314) for the files in the installation package. The workload management properties include workload management class definitions. The installation package also includes class names stored in association with pathnames of executable files from which processes may be executed. Class names may be stored in association with pathnames of executable files in the form of a class assignment table or records for such a table.

Please replace the paragraph beginning at page 13, line 13 of Applicants' original specification, with the following amended paragraph:

The example of FIG. 3 illustrates several ways of storing (326) a class name for the workload management class in association with a pathname for the executable file. Storing (326) a class name for the workload management class in association with a pathname for the executable file may be carried out by storing the class name in a class assignment table (328) established for the purpose of holding pathnames of executables and their associated class names. Alternatively, in the example of FIG. 3, storing (326) a class name for the workload management class in association with a pathname for the executable file may be carried out by storing the class name in the executable file (304) itself, such as, for

example, in an extended attribute on the executable. A further method of storing (326) a class name for the workload management class in association with a pathname for the executable file shown in FIG. 3 is storing the class name in a data structure (303) that represents the executable file in an operating system. Examples of data structures representing executable files in operating system include Unix inodes, File Access Table ("FAT") entries in MSDOS_{TM}, and entries in a Master File Table ("MFT") in Microsoft NT_{TM}.

Please replace the paragraph beginning at page 12, line 21 of Applicants' original specification, with the following amended paragraph:

In the method of FIG. 3, installing (302) software also includes configuring (324) the workload management class (318) in dependence upon the workload management properties (314) and storing (326) a class name (330) of the workload management class (318) in association with a pathname (332) for the executable file (304). That is, in this example, a class (318) is installed at the same time that executables are installed (302) on a file system from an installation package (312)--so that at run time, when a process (308) is executed (306) from the executable file (~~204~~304), a class to which the process can be assigned already exists.

Please replace the paragraph beginning at page 13, line 22 of Applicants' original specification, with the following amended paragraph:

The method of FIG. 3 also includes assigning (310) the process (308) to the workload management class (316). Notice that it is the process that is assigned, according to a process identifier or "PID." An `exec()` provides no return value, because the calling process is replaced entirely with the executed process. A `fork()` call, on the other hand, creates a new process as a duplicate of the calling process and returns the PID of the new process. And an `exec()` call ~~give~~gives its newly created process the PID of the calling process that is replaced by the executed process. An operating system shell that executes a process with an `exec()` call therefore may learn the PID of the executed process by calling `fork()` first to get the PID, and then allowing its duplicate to call `exec()`, wiping out the duplicate and executing a process whose IPID is now known to the shell and therefore to the workload manager. Assigning (310) a process (308) to a workload management class (316) may be carried out through an operating system shell as illustrated in the following exemplary pseudocode: